(12) UK Patent Application (19) GB (11) 2 342 470 (13) A

(43) Date of A Publication 12.04.2000

(21) Application No 9821935.5

(22) Date of Filing 09.10.1998

(71) Applicant(s)
International Business Machines Corporation
(Incorporated in USA - New York)
Armonk, New York 10504, United States of America

(72) Inventor(s)
Gordon John Cockburn
Carlos Francisco Fuente
Andrew Key

(74) Agent and/or Address for Service
R J Burt
IBM United Kingdom Limited, Intellectual Property Department, Mail Point 110, Hursley Park,
WINCHESTER, Hampshire, SO21 2JN,
United Kingdom

(51) INT CL$^7$
G06F 12/02

(52) UK CL (Edition R )
G4A AMX

(56) Documents Cited
EP 0590645 A1 EP 0574884 A1 EP 0430668 A2
WO 94/02898 A1 WO 93/03435 A1
Dr. Dobb's Journal, Vol.23, No.10, October 1998, pages 20ff.

(58) Field of Search
UK CL (Edition Q ) G4A AMA AMB AMX
INT CL$^6$ G06F 12/00 12/02
Selected publications and online: COMPUTER, EDOC, JAPIO, WPI

(54) Abstract Title
**A memory management system and method for a data processing system**

(57) A system and method are disclosed for managing memory for a data processing system. The memory user components and processes have means to reserve, allocate, free and unreserve memory, wherein the order of freeing and unreserving is not fixed, and memory may be unreserved without having been freed. The memory management component or process has means to reclaim such memory as has been unreserved but not yet freed in order to satisfy further memory requests from memory user processes.

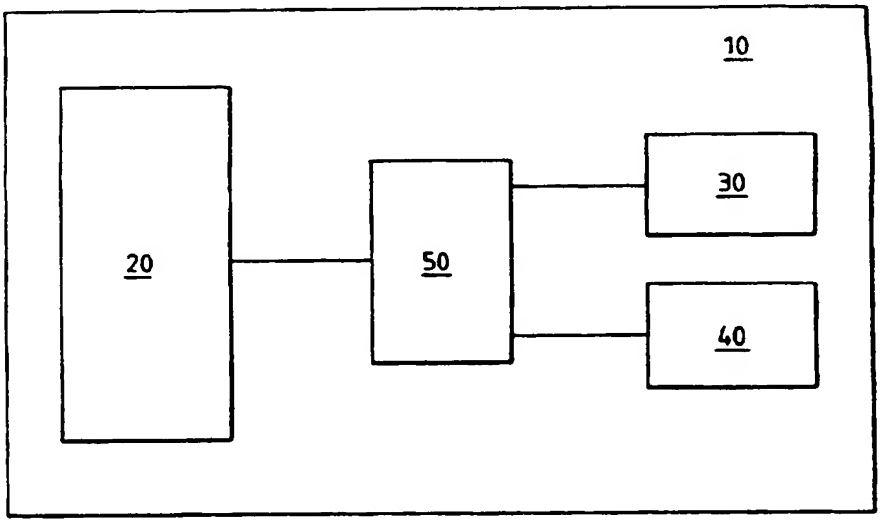


FIG. 1

GB 2 342 470 A

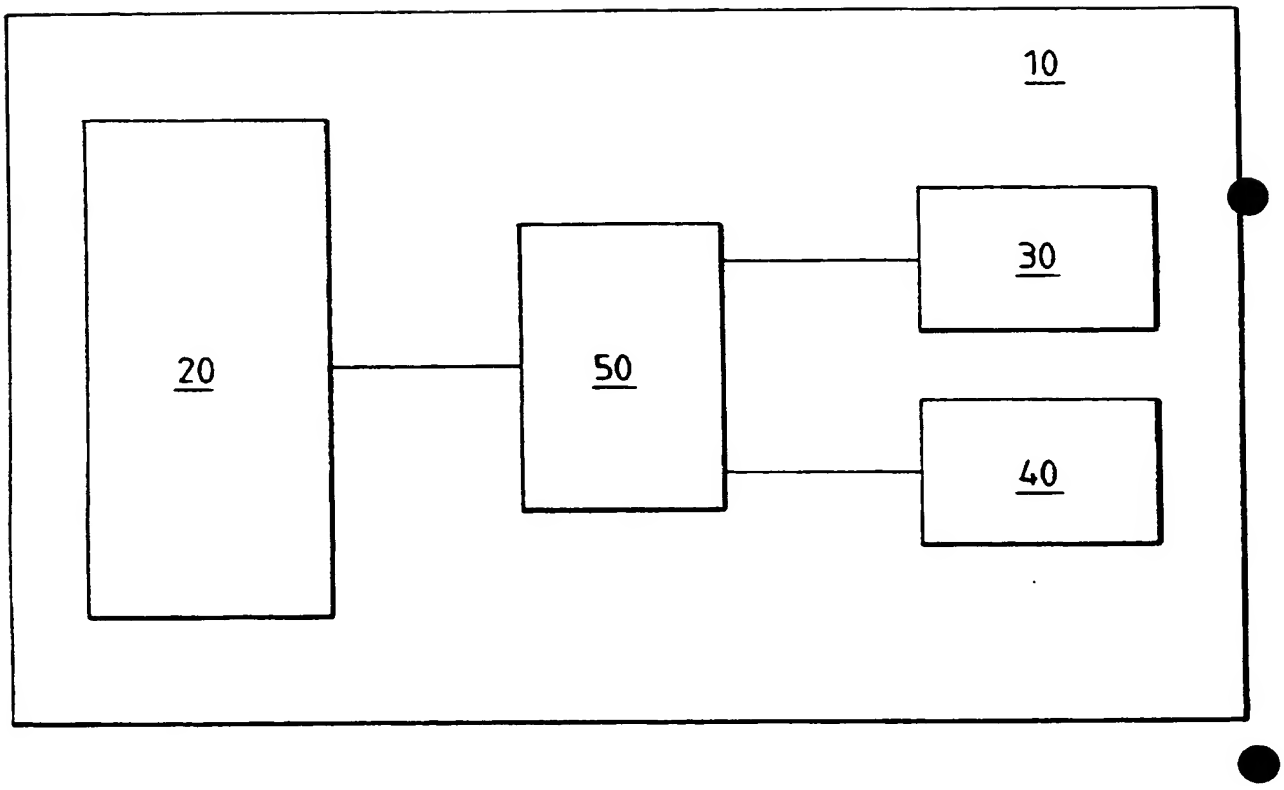


FIG. 1

# A MEMORY MANAGEMENT SYSTEM AND METHOD FOR A DATA PROCESSING SYSTEM

The present invention relates to the field of managing resources in data processing systems, and in particular to managing resources that are scarce or in heavy demand.

In data processing systems, the reserving and allocating of resources, particularly of scarce resources, such as memory, frequently leads to conflict among the tasks and processes as they compete for the use of the resources. For example, processes and tasks require memory for the storage of data and control instructions, and often need to be able to reserve such memory for their anticipated future needs before they are able to begin running, either in part or as a whole.

This conflict is exacerbated in the case of individual devices or components having their own embedded memory, such as device adapters, and handheld, domestic or mobile devices having data processing capabilities and memory, for example. Such devices must typically be manufactured at low cost, and thus cannot have large amounts of embedded memory in case of unanticipated heavy use. They may also need to be made small in order to take up less space in the physical confines of the system or device, and so the space available may determine the amount of embedded memory they may have.

The conflict is further exacerbated in devices used to control complex processing involving large amounts of data. For example, in a device adapter for controlling a Redundant Array of Independent Disks (RAID), the embedded memory is typically heavily used for buffering or caching data, as well as for storing control instructions and retaining parity data for checking that the data distributed about the array has not been corrupted.

Similarly in systems controlling very large databases, the demands on memory in complex processing can be severe, and the performance of the whole system can be affected by conflicts over the use of memory. The same problems may also be encountered in devices or systems for controlling communications, in which large amounts of data may need to be processed and buffered for sending and receiving. Where time constraints on processes also apply, as for example in real-time signal processing and digital image processing, the conflict between the provision of

sufficient memory and the requirement that processes be able to run is particularly acute.

In existing systems, there are various approaches taken to alleviate the problem. Large pools of memory may be statically allocated at system initialisation time, so that the future requirements of the tasks or processes can be met. However, this solution requires that very large amounts of memory are made available to meet all the needs of all the tasks, and this is costly both in terms of the cost of the memory itself, and in terms of the extra space required to accommodate the memory units in the system. In some existing operating systems, there are memory management systems that control memory use dynamically on behalf of the tasks and processes. In such systems, tasks and processes cannot hold onto memory that is not actively in use, and this means that they must issue repeated requests for memory dynamically during task or process execution, even if some of their recently relinquished memory has not been reused in the interim.

Another solution that has been tried is to over-commit memory; that is, for the memory manager to promise to reserve more memory than is present in the system. Subsequently, if a process tries to acquire the memory that it has been promised, the memory manager blocks that process until sufficient memory has become free. This approach has two drawbacks. First, it is possible that the blocked process may be of greater importance than some of the processes that are not blocked, which is an undesirable situation, and second, it is possible for all the processes in the system to become blocked waiting for memory at the same time. In systems that permit memory user processes to dynamically acquire and release memory, programmers often attempt to improve the efficiency of their particular programs by acquiring in advance, and retaining, sufficient memory for any possible later requirement. Such non-altruistic programs have the disadvantage of retaining memory while they do not need it, to the detriment of the other programs in the system.

In conventional systems for managing resources, resource user processes may reserve a resource for future use, then acquire it (or have it allocated to them) for use; they then free it (or have it deallocated), and finally they unreserve it. The order in which these events are permitted is fixed, and this rigidity may contribute to shortages of available memory as described above. In some systems, the

steps of reserving and allocating are combined, as are the steps of freeing and unreserving. Such a system is represented by a conventional pair of malloc and free commands. In such systems, a resource request is made, and if it is successful, the resource is provided immediately.

The present invention accordingly provides a memory management system for a data processing system, comprising reserving means for reserving a first one or more units of memory; allocating means for allocating a second one or more units of memory, where said second one or more units of memory may be some or all of said first one or more units of memory; freeing means for freeing some or all of said second one or more units of memory; unreserving means for unreserving some or all of said first one or more units of memory; and reclaiming means for reclaiming a third one or more units of memory, said third one or more units of memory being units of memory that have been unreserved but not freed.

A desirable aspect of the present invention is the provision of a data processing system comprising a memory management system of the present invention.

A further desirable aspect of the present invention is a device adapter comprising memory controlled by a memory management system of the present invention. Such a device adapter may advantageously be used to control one or more devices, such as storage devices, for example one or more Redundant Arrays of Independent Disks.

A further desirable aspect of the present invention is a portable device, such as a hand-held computer or controller or communications device, comprising local memory controlled by a memory management system of the present invention.

A further desirable aspect of the present invention is an appliance such as a set-top box or other domestic appliance comprising one or more embedded processors and memory controlled by a memory management system of the present invention.

The present invention also provides a method of managing memory in a data processing system, comprising the steps of reserving, for a memory user process or task, a first one or more units of memory; allocating, for a memory user process, a second one or more units of memory, where

said second one or more units of memory may be some or all of said first one or more units of memory; freeing, for a memory user process, some or all of said second one or more units of memory; unreserving, for a memory user process, some or all of said first one or more units of memory; and reclaiming a third one or more units of memory, said third one or more units of memory having been unreserved but not freed.

A preferred embodiment of the present invention will now be described by way of example, with reference to the drawing.

Figure 1 shows a data processing system (10) having memory (20), memory user components or processes (30, 40) and a memory management system (50). The memory user components or processes (30, 40) are arranged to communicate with the memory management system (50) to operate reserving means, allocating means, freeing means and unreserving means provided by the memory management system (50). The memory management system (50) is arranged to manage the memory (20). The memory management system (50) is also arranged to communicate with the memory user components or processes (30, 40) to operate reclaiming means provided by the memory user components or processes (30, 40).

A preferred embodiment will now be further described by way of example wherein the data processing system (10) comprises a device adapter subsystem for controlling a Redundant Array of Independent Disks. Clearly, the use of the memory manager system of the present invention is not limited to such a device adapter subsystem.

In a device adapter subsystem for controlling a Redundant Array of Independent Disks, wherein the device adapter subsystem comprises one or more processors and one or more memories, the memory managed by the memory manager is divided into units of a fixed size. In practice, in a system in which memory is of different types, for example, control memory and data memory, each type is managed independently, but using the same method of management. All memory allocations are performed on a given number of units of the required type, for example, 5 pages of data memory.

Four requests to the memory manager are used by memory user processes. They are:

ReserveXXXX(N) -- this reserves (N) pages of memory of type XXXX. That is, the memory manager checks that there is sufficient memory available to satisfy the request. If there is not sufficient memory available to satisfy the request, the request will fail.

AllocateXXXX() -- this returns a single unit of memory, which must have been previously reserved. This request cannot fail to return the requested memory, because the memory has already been reserved.

FreeXXXX(Y) -- this frees a single unit of memory (Y), which was previously obtained using the Allocate request.

UnreserveXXXX(N) -- this unreserves the specified number of units of memory, which were previously reserved.

Using these requests, a memory user process can determine if enough memory will be available for its use when needed, and can seek a promise that the memory will be reserved for it. However, that memory need not be allocated to that memory user process at the same time that it is reserved. Such reserved, but not yet allocated, memory can be left in an idle cache until it is actually needed.

The memory manager maintains a count of the total units of memory in the system and the total units of memory available for reservation. The memory manager does not allow units of memory to be reserved unless they are available for reservation. It does not over-commit, so that every reservation of a unit of memory is backed up by a unit of memory.

The memory manager does not maintain a count of the number of allocated units of memory, but it does maintain a pool of units of memory that are not yet allocated.

The conventional order of issuing the memory requests is:

1. ReserveXXXX(N)

2. AllocateXXXX()

3. FreeXXXX(Y)

4. UnreserveXXXX(N)

In some circumstances, however, it is advantageous to allow a memory user process to reverse the conventional order of freeing and unreserving units of memory. This is because the memory user might wish to retain some units of memory, even though it does not need them for active processing at the time. An example would be the retention of idle cache memory.

Conventional memory managers do not permit this reversal of the normal order of events, nor do they have the facilities to exploit such a reversal. In the present invention, however, an additional request is available to the memory manager:

ReclaimXXXX(N) -- this enables the memory manager to call on a memory user process to free as many units of memory as it has which are allocated and unreserved, to meet the memory manager's request for (N) units of memory of type XXXX.

The memory user must free the allocated and unreserved memory units immediately; it cannot block the call or delay freeing the memory. Thus, the memory manager has the opportunity to satisfy a further allocate request for a memory user process that needs memory. As stated above, the memory user cannot block or delay the freeing of the memory in response to the Reclaim request. However, in practice, it can select from among its eligible units of memory those which it determines to be least useful to retain, should it have more than one eligible unit of memory.

If the memory user process cannot free enough memory units to satisfy the Reclaim request on its own, the memory manager issues further Reclaim requests to the other memory user processes until the required memory units are freed. Using this facility enables a memory manager to satisfy allocate requests by reusing memory that would otherwise have been used in a cache or some similar idle pool of memory units.

A permitted sequence of issuing memory management requests in an embodiment of the present invention is thus:

1. ReserveXXXX(N) issued by memory user 1.

2. ReserveXXXX(N) issued by memory user 2.

3. AllocateXXXX() issued by memory user 1, acquiring memory unit Y.

4. UnreserveXXXX(N) issued by memory user 1.

5. AllocateXXXX() issued by memory user 2. The memory manager finds it does not have enough free memory units to meet the needs of memory user 2.

6. ReclaimXXXX(N) issued by the memory manager to memory user 1.

7. FreeXXXX(Y) issued by memory user 1. The memory manager meets the needs of memory user 2.

Clearly, the memory manager will not always need to issue a Reclaim request; memory will often be available to satisfy the needs of all the processes in the system.

The above description of the embodiment has been made in terms of requests in a humanly-readable language; it should be clear, however, that such requests may take the form of electronic signal arrangements within the logic of a device or devices.

## CLAIMS

1. A memory management system for a data processing system, comprising:

reserving means for reserving a first one or more units of memory;

allocating means for allocating a second one or more units of memory, where said second one or more units of memory may be some or all of said first one or more units of memory;

freeing means for freeing some or all of said second one or more units of memory;

unreserving means for unreserving some or all of said first one or more units of memory; and

reclaiming means for reclaiming a third one or more units of memory, said third one or more units of memory being units of memory that have been unreserved but not freed.

2. A data processing system comprising a memory management system as claimed in claim 1.

3. A device adapter comprising memory controlled by a memory management system as claimed in claim 1.

4. A device adapter as claimed in claim 3, arranged to control one or more data storage devices.

5. A device adapter as claimed in claim 3, arranged to control a Redundant Array of Independent Disks.

6. A portable device comprising local memory controlled by a memory management system as claimed in claim 1.

7. An appliance comprising one or more embedded processors and memory controlled by a memory management system as claimed in claim 1.

8. A method of managing memory in a data processing system, comprising the steps of:

reserving, for a memory user process or task, a first one or more units of memory;

allocating, for a memory user process, a second one or more units of memory, where said second one or more units of memory may be some or all of said first one or more units of memory;

freeing, for a memory user process, some or all of said second one or more units of memory;

unreserving, for a memory user process, some or all of said first one or more units of memory; and

reclaiming a third one or more units of memory, said third one or more units of memory having been unreserved but not freed.

9. A method of managing memory as claimed in claim 8, wherein the step of freeing selects among said units of memory to be freed.

The Patent Office

10

INVESTOR IN PEOPLE

**Application No:** GB 9821935.5 **Examiner:** David Keston
**Claims searched:** 1-9 **Date of search:** 24 March 1999

## Patents Act 1977
## Search Report under Section 17

**Databases searched:**

UK Patent Office collections, including GB, EP, WO & US patent specifications, in:

UK Cl (Ed.Q): G4A (AMA, AMB, AMX)

Int Cl (Ed.6): G06F 12/00, 12/02

Other: Selected publications and online: COMPUTER, EDOC, JAPIO, WPI

**Documents considered to be relevant:**

| Category | Identity of document and relevant passage | | Relevant to claims |
|---|---|---|---|
| X | EP 0590645 A1 | (MICROSOFT) - see abstract and columns 1 & 2 | 1, 2, 7-9 |
| X | EP 0574884 A1 | (MICROSOFT) - see abstract | 1, 2, 7-9 |
| X | EP 0430668 A2 | (XEROX) - see abstract | 1, 2, 7-9 |
| X | WO 94/02898 A1 | (MICROSOFT) - see abstract | 1, 2, 7-9 |
| X | WO 93/03435 A1 | (PURE SOFTWARE) - see abstract | 1, 2, 7-9 |
| X | Dr. Dobb's Journal, Vol.23, No.10, October 1998, A Petit-Bianco, "Java garbage collection for real-time systems" pages 20ff. | | 1, 2, 7-9 |

| | | | |
|---|---|---|---|
| X | Document indicating lack of novelty or inventive step | A | Document indicating technological background and/or state of the art. |
| Y | Document indicating lack of inventive step if combined with one or more other documents of same category. | P | Document published on or after the declared priority date but before the filing date of this invention. |
| & | Member of the same patent family | E | Patent document published on or after, but with priority date earlier than, the filing date of this application. |